XML 문서에서의 단계화된 스키마 추출

김성림[†] · 윤용익[††]

요 약

인터넷상에서 데이터를 표현하고 교환하는 새로운 표준으로 등장하는 XML 문서는 정해진 스키마를 가지고 있지 않다. XML 문서를 기존의 관계형 데이터베이스나 객체 지향 데이터베이스 질의어에 바로 적용하기에는 부적합하여 이러한 XML 문서에 대해 스키마를 추출하는 방법과 질의어에 대한 연구가 활발히 진행되고 있다. 스키마가 있다면 XML 문서에 대해 사용자 질의를 효율적으로 처리할 수 있다. 그리고 수많은 데이터에서 사용자의 질의에 대한 결과는 너무 많거나 적을 수가 있다. 사용자에게 알맞은 질의 결과를 보여주는 것은 중요하다. 본 논문에서는 XML문서의 엘리먼트 정보를 바탕으로 스키마를 추출하고, 그 발생 빈도 수에 따라 여러 단계의 스키마를 추출하는 방법을 제시하고, 이를 구현하여 그 결과를 분석해본다.

The Levelized Schema Extraction in XML Documents

Sungrim Kim[†] and Yongik Yoon[††]

ABSTRACT

XML documents, which are becoming new standard for expressing and exchanging data in the Internet, don't have defined schema. It is not adequate to directly apply XML documents to the existing SQL or OQL. Research on how to extract schema for XML documents and query language is going on actively. For users' query, the results could be too many or too less. It is important to give the users adequate results. This paper suggests the way to extract many levelized schema according to the frequency of element occurrence in XML documents. The Schema can be reduced or extended to correspond to the users' query more flexibly.

Key words: XML document, schema, frequency

1. 서 론

XML(eXtended Markup Language)은 인터넷상에서 데이터를 표현하고 교환하는 새로운 표준으로 등장하고 있다[4,5]. HTML과 마찬가지로 XML은 SGML의 부분집합이지만 HTML 태그가 데이터 아이템 표현에 중점을 둔 것이라면, XML 태그는 데이터 자체를 기술한다. 따라서 XML은 자기 서술적인 (self-describing) 특징을 바탕으로 XML 문서를 여러 형태로 보여줄 수 있고, 내용을 기반으로 데이터를 필터링하거나 어플리케이션의 목적에 맞게 재구성이 가능하다.

XML 문서는 데이터 구조를 나타내는 중첩된 태그 엘리먼트의 집합으로 구성되고, 기존의 데이터베이스에서처럼 정해진 스키마를 갖고 있지 않지만 문서마다 어떤 구조(Document Type Definition : DTD)를 가지고 있다고 볼 수 있다. 이렇게 XML 데이터 모델은 구조상 기존의 데이터베이스와 많은 차이점이 있어 기존의 관계형 데이터베이스나 객체 지향 데이터베이스 질의어를 바로 적용하기가 힘들다. 따라서 이러한 XML 문서들에 대해 스키마를 추출하는 방법과 질의어에 대한 연구가 활발히 진행되고 있다[1,2,10,12,13,15].

본 논문에서는 XML 문서의 엘리먼트 정보와 그 발생 빈도수를 바탕으로 스키마를 추출하고, 사용자

[†] 정회원, 동덕여자대학교 정보학부 컴퓨터학 전공 강의전임강사
[††] 숙명여자대학교 정보과학부 멀티미디어학과 교수

질의에 대해 엘리먼트의 발생 빈도 수를 조정함으로써 여러 단계의 스키마를 추출하는 방법을 제시하고자 한다. 이러한 스키마 추출 방법은 사용자 질의에 대해 질의 수행 결과가 너무 적거나 많을 때 사용자 질의에 적용되는 스키마를 달리함으로써 질의 범위를 축소 혹은 확장 가능하도록 하기 때문에 사용자의 요구를 효율적으로 반영할 수 있게 한다.

본 논문의 구성은 다음과 같다. 2장에서는 스키마 추출에 대한 관련 연구를 살펴보고, 3장에서는 본 논문의 배경이 되는 이론과 몇 가지 정의를 서술한다. 4장에서는 본 논문에서 제안하고자 하는 스키마를 추출하는 알고리즘과 예제를 보이고, 5장에서 스키마 추출을 위해 제안한 방법을 실험한 내용을 설명하고, 6장에서 결론을 맺는다.

2. 관련 연구

트리 표현식들의 발생 빈도에 따라 최대의 트리 표현식으로 공통적인 스키마를 추출하는 방법이 있다[16,17]. 트리 표현식(*te* : tree expression) 에서 *te*의 지지도(support : MINISUP)는 도큐먼트 *d* 보다 표현식이 약한 *te*를 갖는 도큐먼트의 개수이다. MINISUP보다 큰 지지도를 가지면서 가장 많은 정보를 포함하는 트리 표현식이 추출되는 스키마가 된다. 자주 발생되는 비슷한 질의에 대해서 트리 표현식을 만들고, 이를 바탕으로 스키마를 추출함으로써 유사한 질의에 대해 효율적으로 실행할 수 있다는 장점이 있지만 문서 전체에 대한 스키마를 찾기 힘들다는 제약이 있다.

발생 빈도 패턴을 찾는 방법은 트랜잭션 데이터베이스, 시계열 데이터베이스등 많은 데이터베이스 분야에서 연구되어 왔다. 여러 방법 중에서 발생 빈도 패턴 트리(*FP-tree* : Frequent Pattern Tree)를 구축하여 최대 패턴을 구하는 방법이 제시되었다[8]. FP-tree에서는 발생 빈도 패턴에 대한 정보를 저장하고, 이를 이용하여 *조건 FP-tree*를 형성함으로써 빈도 패턴을 찾아내는데 보다 효율성을 증가시켰다. 사용자 정의 발생 빈도 수를 바탕으로 스키마가 다양하게 추출될 수 있다는 장점은 있지만 전체 스키마보다는 특정 패턴에 대한 스키마가 추출되는 제약과 특정 패턴과 발생 빈도 수에 대한 사용자 정의 값에 대해 매번 스키마 추출 단계를 반복해야 한다는 제약점이 있다.

Lore는 스탠포드대학에서 개발한 XML을 위한 데이터베이스 관리 시스템이다[3,6]. Lore는 XML 데이터의 구조를 파악하기 위해 DataGuide를 제공한다. DataGuide는 XML 데이터베이스에 대해 정확하고, 동적으로 정리된 구조를 표현해줌으로써 데이터베이스 스키마나 DTD 역할을 수행하게 된다. 사용자는 DataGuide를 통해 데이터베이스의 전체적인 구조를 파악하여 질의를 만들 수 있게 된다. DataGuide는 모든 문서에 대한 전체적인 스키마를 파악할 수 있다는 장점이 있지만 모든 문서에 있는 데이터가 표현됨으로써 생성되는 스키마의 범위가 최대가 되고 따라서 질의 수행을 위한 검색 범위가 넓어질 수 있다는 제약점이 있다.

3. 모델링

본 장에서는 본 논문에서 제안하는 스키마 추출 방법에 필요한 기본 개념을 살펴보고, 몇 가지 정의를 설명한다.

3.1 Edge Labeled Graph

XML 문서를 반구조적 데이터(semi-structured data)처럼 방향성있는 edge-labeled graph로 표현할 수 있다[5,7]. Edge-labeled graph에서 엘리먼트는 객체(노드)로 표현되고, 각 객체는 객체 식별자 *oid* (object identifier)를 갖고 두 개의 객체-단순 객체와 복합 객체-로 구분된다. Edge-labeled graph에서는 객체들간에 간선이 존재하고, 각 간선마다 엘리먼트 이름으로 레이블이 있고, 서브 엘리먼트를 표현하는 방향성을 갖는다.

3.2 스키마 추출을 위한 그래프

스키마 추출을 위하여 본 논문에서는 다음과 같이 두 개의 그래프를 정의한다.

ㅇ 정의 1: *데이터 그래프 (Data Graph)*

XML 문서의 모든 데이터가 표현되는 edge labeled directed graph를 '*데이터 그래프(Data Graph)*'라고 정의한다. 루트 노드로부터 하위 노드로 방향성 있는 간선이 만들어지고, 간선의 레이블은 엘리먼트의 이름이 된다. 그리고 각 노드는 oid를 갖고, 노드는

단순 객체 또는 복합 객체의 형태를 갖는다.

ㅇ 정의 2: *스키마 그래프 (Schema Graph)*

데이터 그래프에서 깊이 우선 탐색 기법을 바탕으로 모든 경로가 단 한번만 표현되는 그래프를 '*스키마 그래프(Schema Graph)*'라고 정의한다. Lore 시스템[11]의 DataGuide[7]처럼 스키마 그래프에서는 모든 레이블 경로가 유일하고(concise), XML 문서에 있는 모든 데이터는 표현되어야 하고(accuracy), 각 노드의 구성이 어떻게 되어있는지(convenience) 알 수 있도록 한다.

3.3 비트맵 인덱싱을 이용한 레이블 경로 인덱싱

비트맵 인덱싱의 기본 개념은 튜플에 있는 애트리뷰트가 어떤 특별한 값을 갖고 있느냐 없느냐를 0 혹은 1의 비트로 표현하는 것이다[9,14]. 본 논문에서는 아래와 같이 정의한 XML 문서에서의 레이블 경로를 비트맵 인덱싱하여 보다 유동적인 스키마 그래프를 생성한다[18].

ㅇ 정의 3: *레이블 경로 (Label Path)*

데이터 그래프 혹은 스키마 그래프에서 한 노드에서 어떤 하위 노드로의 경로를 '*레이블 경로(Label Path)*' 라고 정의한다. 그래프에서 노드와 노드사이에는 간선이 존재하고, 간선은 엘리먼트 이름으로 레이블이 존재한다. 그리고 루트 노드에서 리프 노드까지의 경로에서 나타나는 중간 노드는 . (dot)을 사용하여 표현한다.

4. 스키마 추출

본 장에서는 3장에서 설명한 기본 개념을 바탕으로 영화에 대한 정보를 표현하는 XML문서를 예로 들어 설명한다.

4.1 예제 XML 문서

웹 사이트 http://www.imdb.com/에 있는 영화 정보에 대한 내용 중에서 Top250flims에서 100위까지의 멀티미디어 데이터를 제외한 텍스트 데이터를 바탕으로 생성한 XML 문서를 가정한다[19]. 생성한 XML 문서의 예는 그림 1과 같다.

4.2 데이터 그래프와 스키마 그래프

그림 1의 예제 XML 문서를 데이터 그래프로 표현하면 그림 2와 같다. 3.2에서 정의한바와 같이 그림 2의 데이터 그래프는 예제 XML 문서에 있는 모든 엘리먼트가 표현되어 있다.

```
<movie>
  <title>Citizen Kane </title>
  <year>1941</year>
  <director>Orson Welles</director>
  <writer>Herman J. Mankiewicz  </writer>
  <writer><firstname>Orson</firstname> <lastname>Welles</lastname></writer>
  <genre>Drama</genre>
  <cast>
     <name>Orson Welles</name><role>Charles Foster Kane </role>
     <award>Oscar</award>
     <category>Best Writing, Original Screenplay</category>
     <spouse><name>Rita Hayworth </name> <occupation>divorced</occupation></spouse>
  </cast>
  <cast>
     <name>Dorothy Comingore</name> <role>Susan Alexander Kane </role>
     <spouse><name>Richard Collins (I) </name> <occupation>?</occupation></spouse>
  </cast>
  <language>English</language> <country>USA</country> <color>Black and White</color>
  <keywords>sled</keywords> <keywords>dying-words  </keywords>
</movie>
```

그림 1. 영화 정보에 대한 예제 XML문서 (d1)
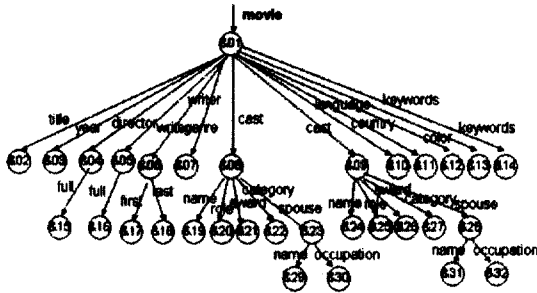
그림 2. XML 문서 (d1)에 대한 데이터 그래프

그림 1의 예제 XML 문서를 스키마 그래프로 표현하면 그림 3과 같다. 3.2에서 정의한바와 같이 그림 3의 스키마 그래프는 모든 레이블 경로가 단 한번씩만 표현된다.
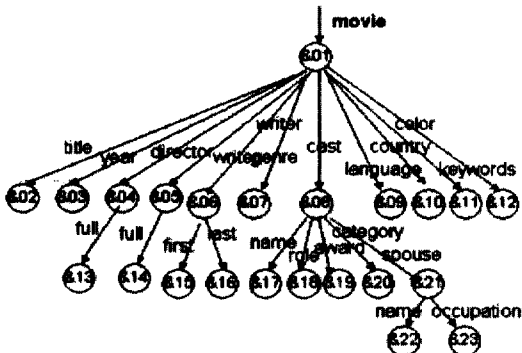
그림 3.XML 문서(d1)에 대한 스키마 그래프

4.3 레이블 경로 비트맵 인덱싱을 이용한 스키마 추출

4.3.1 레이블 경로

스키마 그래프에 대해 레이블 경로는 루트 노드부터 리프 노드까지 깊이 우선 탐색 기법을 바탕으로 중간노드의 레이블이 레이블 경로에 추가되는데 그 방법은 알고리즘 4.1과 같다. 레이블 경로는 리프 노드 개수만큼 구해지고, 중복되는 레이블 경로는 존재하지 않는다. 비트맵 인덱스 기법을 그림 3의 스키마 그래프에서 레이블 경로에 적용하면 각 XML 문서에서의 레이블 경로의 존재여부에 따라 1 혹은 0의 값을 갖는다. 이는 알고리즘 4.2에 기술되어 있다.

4.3.2 레이블 경로 빈도 수를 이용한 스키마 추출

모든 XML 문서에 대한 비트맵 인덱스에 대해 bitwise-OR 연산을 수행하면 모든 레이블 경로가 포함되는 스키마 그래프가 생성되고, 이 그래프는 데이터 그래프에서 생성되는 스키마 그래프와 동일하다. 이는 알고리즘 4.3과 같다.

모든 XML 문서에서 각 레이블 경로에 대해 bitwise-OR 연산을 수행하면 모든 레이블 경로가 표현되는 광범위한 스키마 그래프를 구할 수 있고, bitwise-AND 연산을 수행하면 모든 XML 문서에서 공통적으로 존재하는 레이블 경로로만 구성된 스키마 그래프가 생성되어 질의 범위를 축소할 수 있다. XML 문서의 각 레이블 경로의 발생 빈도 수를 조절하여 스키마 그래프를 생성함으로써 그 질의 범위를 유동적으로 할 수 있다. 여러 단계의 스키마를 추출하기 위해 레이블 경로의 발생 빈도 수를 알고리즘 4.4에 의해 계산한다.

알고리즘 4.1. 레이블 경로 구하기

```
MakeLabelPath (treeNode u)
 begin
    path = u.label
    while (u isNot leafNode)
     begin
        v = u.childNode
        path = path + v.label
        u = v
     end
 end
```

알고리즘 4.2 레이블 경로 인덱싱

```
CalcBitVector(document d)
begin
    while (pi isNot EOF)
        if (pi in d)
           d(pi) = 1
        else
           d(pi) = 0
        endif
end
```

알고리즘 4.3 레이블 경로에 대한 bitwise-OR 연산

```
// pi : 레이블 경로, di : XML 문서
bitwiseORLabelPath()
 begin
    for each i in path P
      new_path(i) = bitwise_OR(ΣdjPi)
end
```

계산된 레이블 경로 발생 빈도 수에 사용자 정의 임계치를 적용하면 임계치 이상인 레이블 경로로만 구성되는 스키마 그래프를 구할 수 있다. 그 방법은 알고리즘 4.5와 같다.

알고리즘 4.4 레이블 경로 빈도수 계산

```
// di : XML 문서
countLabelPath()
 begin
    for each i in path P
      Freq(i) = count(Σdj)
 end
```

알고리즘 4.5 임계치에 따른 레이블 경로 비트맵

```
FilterFreq(threshold t)
 begin
    while (Freq(i) isNot EOF)
       if (Freq(i) >= t)
          New_Freq(i) = 1
       else
          New_Freq(i) = 0
       end if
 end
```

5. 실 험

5.1 실험 환경

본 논문에서 제시한 레이블 경로의 발생 빈도 수에 따른 스키마 추출 방법의 실험 환경은 다음과 같다. 운영 체제는 Windows2000이고, 데이터베이스는 Oracle9i 를 사용하였다. 구현 언어는 JDK 1.3.1 과 JSP를 사용하였고, 오라클과의 연동을 위하여 Oracle JDBC thin driver 를 사용하였다. 웹서버는 IIS 5.0, JSP 엔진으로는 resin 2.0.1을 사용하고, XML 파서로는 Oracle Parser (버전 2.0.1.0) 를 사용하였다.

5.2 실험 모델

본 논문에서 사용된 XML 문서는 영화에 관한 정보가 있는 http://www.imdb.com를 참조하였다[19]. Top250 films 중에서 100위까지의 데이터를 멀티미디어 데이터를 제외한 텍스트 데이터를 중심으로 XML 문서를 생성하여 실험하였다.

5.3 실험

5.3.1 데이터 그래프 생성 결과

그림 4는 입력 XML 문서들에 대해 파싱 작업을 하고 생성된 데이터 그래프가 데이터베이스에 저장된 결과의 일부이다. 각 XML 문서마다 doc_id 가 부여되고, 루트 노드부터 리프 노드까지 레이블 경로가 생성되고, 이 레이블 경로에 대해 리프 노드의 데이터가 저장되고 유일한 id를 갖는다.

5.3.2 스키마 그래프 생성 결과

레이블 경로가 유일하게 표현될 수 있도록 스키마 그래프를 생성한다. 그림 5는 데이터베이스에 저장된 스키마 그래프의 결과이다.

그림 4. 데이터베이스에 저장된 데이터 그래프

그림 5. 데이터베이스에 저장된 스키마 그래프

5.3.3 레이블 경로 발생 빈도수 계산 결과

스키마 그래프를 바탕으로 데이터 그래프에 저장된 각 문서의 레이블 경로를 적용하여 레이블 경로가 존재하면 1, 존재하지 않으면 0으로 표현하도록 하였다. 그림 6은 스키마 그래프와 데이터 그래프를 이용하여 각 XML 문서를 비트 벡터로 표현한 결과이다.

그림 6. 레이블 경로의 비트맵 인덱싱

그림 7은 레이블 경로의 총 발생 빈도 수를 계산한 것이다. 그림 7의 결과를 보면 cast_name, cast_role, genre, title, writer에 대한 레이블 경로가 100개로 실험한 모든 XML 문서에 나타난다는 것을 알 수 있고, director의 레이블 경로는 FirstName과 LastName으로 구별되어 표현되고, 49개의 최소 발생 빈도수임을 알 수 있다.

5.4 레이블 경로의 발생 빈도 수에 따라 생성되는 스키마

레이블 경로 발생 빈도 수에 대한 정보를 파악한 후 임의의 임계치를 부여하여 질의 처리 전에 그 질의 범위를 축소 혹은 확대 할 수 있다. 레이블 경로 발생 빈도수에 대한 임계치는 0.0에서 1.0 사이의 값을 입력받도록 하였다. 즉, 총 XML 문서의 개수와 레이블 경로 발생 빈도 수를 0.0과 1.0 사이로 정규화시켜 적용시켰다. 임의로 임계치를 0.5와 0.8로 했을 때 추출되는 스키마(레이블 경로)와 적용되는 XML 문서의 번호와 개수가 그림 8, 9와 같은 결과를 보여준다.

5.5 실험 결과 분석

임계치가 0.5 이상인 경우는 총 XML 문서 100개 중에서 발생 빈도수가 50개이상인 레이블 경로만을 추출하여 스키마를 생성하게 된다. 임계치가 0.8 이상인 경우는 총 XML 문서 100개 중에서 발생 빈도수가 80개이상인 레이블 경로만을 추출하여 스키마를 생성한다. 임계치가 0.5인 경우와 0.8인 경우에 추출되는 스키마의 형태를 비교해보면 0.8인 경우에 추출되는 스키마에 대해서는 시나리오 작가의 정보(writer)

그림 7. 레이블 경로 발생 빈도수

그림 8. 임계치 >= 0.5

그림 9. 임계치 >= 0.8

를 알 수 있는 질의문을 수행할 수 없지만 임계치가 0.5인 경우에 추출되는 스키마에 대해서는 질의문을 수행할 수 있다는 것을 알 수 있다.

임계치가 0.5인 경우는 추출되는 스키마는 임계치가 0.8인 경우보다 많은 레이블 경로를 포함하지만 스키마에 나타나는 모든 레이블 정보를 포함하는 XML 문서의 개수는 9개로써 전체 문서의 10%도 안되는 범위임을 알 수 있다. 하지만 임계치가 0.8인 경우는 추출되는 스키마의 범위는 0.5인 경우보다는 좁지만 스키마에 나타나는 모든 레이블 정보를 포함하는 XML 문서의 개수는 86개로써 전체 문서의 86%임을 알 수 있다. 따라서 사용자 질의가 임계치 0.5인 경우에 생성되는 스키마에 적용될 수 있다면 질의가 처리되기 전에 XML 문서의 검색 범위를 미리 줄일 수 있을 것이다. 따라서 본 실험을 통해 사용자의 질의에 대해 각각 다른 스키마에 적용을 시키면 질의 수행에 대해 문서의 범위도 유동적일 수 있음을 알 수 있다.

6. 결 론

XML이 인터넷상에서 데이터를 표현하고 교환하는 새로운 표준으로 등장하고 있다. XML은 미리 정의된 스키마가 없고, 문서 자체에 데이터와 데이터 구조를 갖고 있기 때문에 기존의 관계형 데이터베이스나 객체 지향 데이터베이스에서 사용되는 SQL이나 OQL을 바로 적용하기가 어렵다. 따라서 이러한 XML에 대해 새로운 질의어와 질의 처리를 위한 스키마 추출에 대한 많은 연구가 이루어지고 있다.

본 논문에서는 XML 문서에 대해 레이블 경로의 발생 빈도수에 따른 스키마 추출 방법을 제안하였다. 스키마 추출 방법은 일단 같은 DTD를 갖는 XML 문서들에 대해 스키마 그래프를 생성하여 XML 문서에 있는 모든 엘리먼트의 정보가 모두 표현되면서 단 한번만 표현될 수 있도록 한다. 그리고 스키마 그래프를 바탕으로 입력 XML 문서의 레이블 경로 존재 여부를 비트 벡터로 표현하고, XML 문서에서 레이블 경로의 발생 빈도 수를 계산하였다. 그리고 어떤 임계치에 따라 여러 단계의 스키마 추출을 가능하게 함으로써 사용자 질의에 대해 보다 효율적으로 처리할 수 있도록 하였다.

본 논문에서 제안하는 방법은 XML 문서에 나타나는 레이블 경로의 발생 빈도수에 따라 여러 단계의 스키마 추출을 가능하게 함으로써 사용자 질의에 대해 보다 유동적으로 적용시킬 수 있음을 알 수 있다. 사용자 질의에 대해 너무 적은 혹은 너무 많은 질의 결과가 나왔다면 임계치를 두어 그 질의 범위를 축소 혹은 확장하여 사용자 질의에 보다 적합한 결과를 보여줄 수 있다.

향후 연구과제로는 메타 데이터를 이용한 스키마 추출방법을 고려하여 데이터의 형태뿐만 아니라 의미적으로도 분석이 가능하게 함으로써 보다 효율적인 스키마를 추출하는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] Serge Abiteboul, “Querying Semi-Structured Data”, In IDCT 1997

[2] Serge Abiteboul, Peter Buneman, Dan Suciu, “Data on the Web : From Relations to Semistructured Data and XML”, Morgan Kaufmann, 2000

[3] S. Abiteboul, D. Quass, J. McHugh, J. Widom and J. Wiener, “The Lorel Query Language for Semistructred Data”, International Journal of

Digital Libraries, 1(1):66-88, 1997

[4] Jon Bosak, "XML, Java, and the Future of the Web", http://webreview.com/wr /pub/97/12/19/xml/index.html

[5] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0", http://www.w3.org/TR/REC-xml#dt-xml-doc, 1998

[6] Roy Goldman, Jason McHugh, Jennifer Widom, "Lore: A Database Management System for XML", Dr. Dobb's Journal, 25(4):76-80. April 2000, http://www.ddj.com/articles/2000/0004/0004i/0004i. htm ?topic=xml

[7] Roy Goldman, Jennifer Widom, "DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases", In Proceedings of VLDB, 1997

[8] Jiawei Han, Jian Pei, Yiwen Yin, "Mining Frequent Patterns without Candidate Generation", Proceedings of the 2000 ACM SIGMOD on Management of data, 2000, pp.1-12

[9] Theodore Johnson, "Performance Measurements of Compressed Bitmap Indices", VLDB 1999, pp. 278-289

[10] Alon Levy, "More on Data Management for XML", University of Washington, May 9th, 1999. http://www.cs.washington.edu/homes/alon/widom-response.html

[11] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, J. Widom, "Lore : A Database Management System for Semistructured Data", SIGMOD Record, 26(3), pp.54-66, September 1997

[12] Jayavel Shanmugasundaran, Kristin Tufte, Gang He, Chun Zhang, David DeWit, Jeffrey Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities", Proceedings of the 25th VLDB Conference 1999

[13] Dan Suciu, "Semistructured Data and XML", In Proceedings of International Conference on Foundation of Data Organization, 1998

[14] M.C. Wu, A.P. Buchmann, "Encoded Bitmap Indexing for Data Warehouses", Proc. ICDE '98, 220-230

[15] Jennifer Widom, "Data Management for XML", Working Document, initial draft appeared April 1999, Also IEEE Data Engineering Bulletin, Special Issue on XML, 22(3):44-52, September 1999.

[16] Ke Wang, Huiqing Liu, "Schema Discovery from Semistructured Data", International Conference on Knowledge Discovery and Data Mining, August 1997, pp.271-274

[17] Ke Wang, Huiqing Liu, "Discovering Typical Structures of Documents : A Road Map Approach", The ACM SIGR conference on Research and Development in Information Retrieval, August 1998, pp.146-154

[18] J. Yoon, S. Kim, "Schema Extraction for Multimedia XML Document Retrieval", in Proc. of International Database Symposium on Mobile, XML and Post-Relational Databases, Hong Kong, June 2000. Also to appear in Journal of Applied Systems Studies, Cambridge International Science Publishing, Cambridge, UK, 2001

[19] http://us.imdb.com/top_250_films/

김 성 림

1994년 숙명여자대학교 전산학과 졸업 (이학사)
1997년 숙명여자대학교 대학원 전산학과 졸업 (이학 석사)
2002년 숙명여자대학교 대학원 컴퓨터학과 졸업 (이학 박사)
2001년~현재 동덕여자대학교 정보학부 컴퓨터학전공 강의전임강사
관심분야 : XML, 멀티미디어 데이터베이스, 질의 처리

윤 용 익

1983년 동국대학교 통계학과 졸업 (이학사)
1985년 한국과학기술원 전산학과 졸업 (공학 석사)
1994년 한국과학기술원 전산학과 졸업 (공학 박사)
1997년~현재 숙명여자대학교 정보학부 멀티미디어학과 교수
관심분야 : 정보통신, 멀티미디어 통신, 분산 시스템, 실시간 처리 시스템, 분산 미들웨어 시스템, 분산 데이터베이스 시스템, 실시간 OS/DBMS